Quantum machine learning

András Gilyén

Alfréd Rényi Institute of Mathematics
Budapest, Hungary

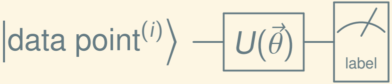
Quantum Computing Summer School, Physikzentrum Bad Honnef, Germany
2022 August 14-19

Major families of quantum machine learning algorithms

- "Quantum neural networks" (i.e., variational quantum circuits)

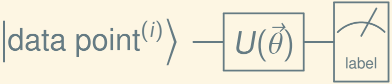
Major families of quantum machine learning algorithms

- "Quantum neural networks" (i.e., variational quantum circuits)
  –For example classification (supervised learning)



Major families of quantum machine learning algorithms

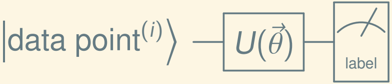
- "Quantum neural networks" (i.e., variational quantum circuits)
  –For example classification (supervised learning)



Optimize parameters to try and find most accurate model

Major families of quantum machine learning algorithms

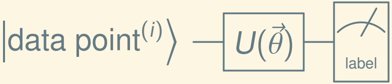
- "Quantum neural networks" (i.e., variational quantum circuits)
  –For example classification (supervised learning)



Optimize parameters to try and find most accurate model
–How to avoid barren plateaus?

Major families of quantum machine learning algorithms

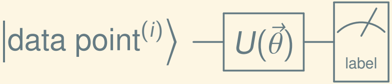
- "Quantum neural networks" (i.e., variational quantum circuits)
  –For example classification (supervised learning)



Optimize parameters to try and find most accurate model
–How to avoid barren plateaus?
–Small scale experiments don't provide conclusive evidence, time will tell ...

Major families of quantum machine learning algorithms

- “Quantum neural networks” (i.e., variational quantum circuits)
  –For example classification (supervised learning)

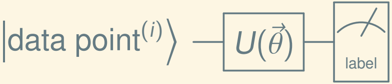


  Optimize parameters to try and find most accurate model
  –How to avoid barren plateaus?
  –Small scale experiments don’t provide conclusive evidence, time will tell . . .

- “Big Data” analysis via quantum linear algebra methods

Major families of quantum machine learning algorithms

- “Quantum neural networks” (i.e., variational quantum circuits)
  –For example classification (supervised learning)

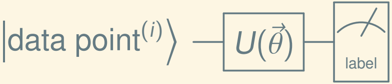


  Optimize parameters to try and find most accurate model
  –How to avoid barren plateaus?
  –Small scale experiments don’t provide conclusive evidence, time will tell . . .

- “Big Data” analysis via quantum linear algebra methods
  –For example least squares regression (via quantum matrix inversion, i.e., HHL)

Major families of quantum machine learning algorithms

- “Quantum neural networks” (i.e., variational quantum circuits)
  –For example classification (supervised learning)



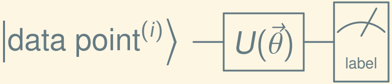
  Optimize parameters to try and find most accurate model
  –How to avoid barren plateaus?
  –Small scale experiments don’t provide conclusive evidence, time will tell . . .

- “Big Data” analysis via quantum linear algebra methods
  –For example least squares regression (via quantum matrix inversion, i.e., HHL)

- Speeding up optimization (learning) with quantum algorithms

Major families of quantum machine learning algorithms

- "Quantum neural networks" (i.e., variational quantum circuits)
  –For example classification (supervised learning)



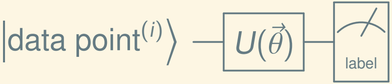
  Optimize parameters to try and find most accurate model
  –How to avoid barren plateaus?
  –Small scale experiments don't provide conclusive evidence, time will tell . . .

- "Big Data" analysis via quantum linear algebra methods
  –For example least squares regression (via quantum matrix inversion, i.e., HHL)

- Speeding up optimization (learning) with quantum algorithms
  –For example quantum linear program (LP) and SDP solving

Major families of quantum machine learning algorithms

- "Quantum neural networks" (i.e., variational quantum circuits)
  –For example classification (supervised learning)



  Optimize parameters to try and find most accurate model
  –How to avoid barren plateaus?
  –Small scale experiments don't provide conclusive evidence, time will tell . . .

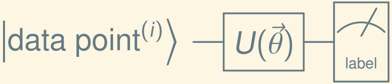
- "Big Data" analysis via quantum linear algebra methods
  –For example least squares regression (via quantum matrix inversion, i.e., HHL)

- Speeding up optimization (learning) with quantum algorithms
  –For example quantum linear program (LP) and SDP solving

- Learning from quantum data

Major families of quantum machine learning algorithms

- "Quantum neural networks" (i.e., variational quantum circuits)
  –For example classification (supervised learning)



  Optimize parameters to try and find most accurate model
  –How to avoid barren plateaus?
  –Small scale experiments don't provide conclusive evidence, time will tell . . .

- "Big Data" analysis via quantum linear algebra methods
  –For example least squares regression (via quantum matrix inversion, i.e., HHL)

- Speeding up optimization (learning) with quantum algorithms
  –For example quantum linear program (LP) and SDP solving

- Learning from quantum data
  –Understanding properties of a quantum state or a quantum process

Quantum machine learning for Big Data

Some major tasks, given data $A \in \mathbb{R}^{m \times n}$

- Principal component analysis: find large eigenvalues and eigenvectors
  Quantum: Lloyd, Mohseni, and Rebentrost 2013

Quantum machine learning for Big Data

Some major tasks, given data $A \in \mathbb{R}^{m \times n}$

- Principal component analysis: find large eigenvalues and eigenvectors
  Quantum: Lloyd, Mohseni, and Rebentrost 2013
- Supervised clustering: given class labels, classify new data (vector) b
  Quantum: Lloyd, Mohseni, and Rebentrost 2013

Quantum machine learning for Big Data

Some major tasks, given data $A \in \mathbb{R}^{m \times n}$

- Principal component analysis: find large eigenvalues and eigenvectors
  Quantum: Lloyd, Mohseni, and Rebentrost 2013
- Supervised clustering: given class labels, classify new data (vector) b
  Quantum: Lloyd, Mohseni, and Rebentrost 2013
- Support vector machines: binary classification of data (vector) b
  Quantum: Rebentrost, Mohseni, and Lloyd 2013

Quantum machine learning for Big Data

Some major tasks, given data $A \in \mathbb{R}^{m \times n}$

- Principal component analysis: find large eigenvalues and eigenvectors
  Quantum: Lloyd, Mohseni, and Rebentrost 2013
- Supervised clustering: given class labels, classify new data (vector) b
  Quantum: Lloyd, Mohseni, and Rebentrost 2013
- Support vector machines: binary classification of data (vector) b
  Quantum: Rebentrost, Mohseni, and Lloyd 2013
- Recommendation Systems: find low-rank $A_\sigma \in \mathbb{R}^{m \times n}$ s.t. $A \approx A_\sigma$
  Quantum: Kerenidis and Prakash 2016

Quantum machine learning for Big Data

Some major tasks, given data $A \in \mathbb{R}^{m \times n}$

- Principal component analysis: find large eigenvalues and eigenvectors
  Quantum: Lloyd, Mohseni, and Rebentrost 2013
- Supervised clustering: given class labels, classify new data (vector) b
  Quantum: Lloyd, Mohseni, and Rebentrost 2013
- Support vector machines: binary classification of data (vector) b
  Quantum: Rebentrost, Mohseni, and Lloyd 2013
- Recommendation Systems: find low-rank $A_\sigma \in \mathbb{R}^{m \times n}$ s.t. $A \approx A_\sigma$
  Quantum: Kerenidis and Prakash 2016

Idea: Quantum computers can work with exponentially large matrices and vectors!

Quantum machine learning for Big Data

Some major tasks, given data $A \in \mathbb{R}^{m \times n}$

- Principal component analysis: find large eigenvalues and eigenvectors
  Quantum: Lloyd, Mohseni, and Rebentrost 2013
- Supervised clustering: given class labels, classify new data (vector) b
  Quantum: Lloyd, Mohseni, and Rebentrost 2013
- Support vector machines: binary classification of data (vector) b
  Quantum: Rebentrost, Mohseni, and Lloyd 2013
- Recommendation Systems: find low-rank $A_\sigma \in \mathbb{R}^{m \times n}$ s.t. $A \approx A_\sigma$
  Quantum: Kerenidis and Prakash 2016

Idea: Quantum computers can work with exponentially large matrices and vectors!

Warning: "Read the fine print", Scott Aaronson, Nature, 2015

- Need to be able to efficiently prepare the input vector $|b\rangle$

Quantum machine learning for Big Data

Some major tasks, given data $A \in \mathbb{R}^{m \times n}$

- Principal component analysis: find large eigenvalues and eigenvectors
  Quantum: Lloyd, Mohseni, and Rebentrost 2013
- Supervised clustering: given class labels, classify new data (vector) b
  Quantum: Lloyd, Mohseni, and Rebentrost 2013
- Support vector machines: binary classification of data (vector) b
  Quantum: Rebentrost, Mohseni, and Lloyd 2013
- Recommendation Systems: find low-rank $A_\sigma \in \mathbb{R}^{m \times n}$ s.t. $A \approx A_\sigma$
  Quantum: Kerenidis and Prakash 2016

Idea: Quantum computers can work with exponentially large matrices and vectors!

Warning: "Read the fine print", Scott Aaronson, Nature, 2015

- Need to be able to efficiently prepare the input vector $|b\rangle$
- Need a circuit implementation (block-encoding) of the input matrix A

Quantum machine learning for Big Data

Some major tasks, given data $A \in \mathbb{R}^{m \times n}$

- Principal component analysis: find large eigenvalues and eigenvectors
  Quantum: Lloyd, Mohseni, and Rebentrost 2013
- Supervised clustering: given class labels, classify new data (vector) b
  Quantum: Lloyd, Mohseni, and Rebentrost 2013
- Support vector machines: binary classification of data (vector) b
  Quantum: Rebentrost, Mohseni, and Lloyd 2013
- Recommendation Systems: find low-rank $A_\sigma \in \mathbb{R}^{m \times n}$ s.t. $A \approx A_\sigma$
  Quantum: Kerenidis and Prakash 2016

Idea: Quantum computers can work with exponentially large matrices and vectors!

Warning: "Read the fine print", Scott Aaronson, Nature, 2015

- Need to be able to efficiently prepare the input vector $|b\rangle$
- Need a circuit implementation (block-encoding) of the input matrix A
- Need to efficiently extract "answer" from the output $|x\rangle (= A^{-1}|b\rangle)$

Recommendation systems – Netflix challange

	Inside Out	Good Will Hunting	Mean Girls	Terminator	Titanic	Warrior
Tina Fey	3	1	5	1	?	1
Helen Mirren	2	?	?	2	5	1
Sylvester Stallone	1	3	1	4	2	5
Tom Hanks	?	3	1	?	4	3
George Clooney	2	2	1	3	1	4

Image source: https://towardsdatascience.com ©

The assumed structure of preference matrix:

Movies: a linear combination of a small number of features
User taste: a linear weighing of the features

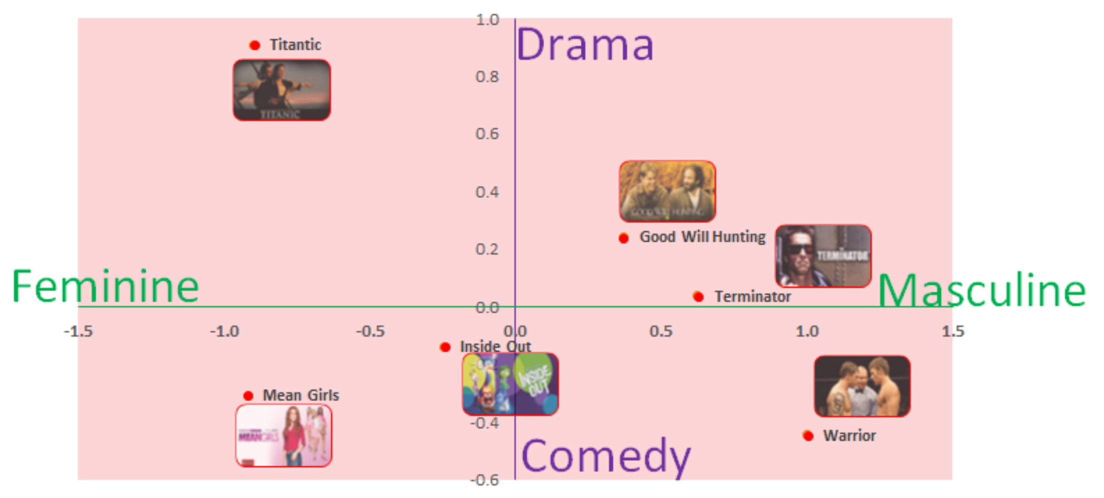


Image source: https://towardsdatascience.com ©

What about the missing data points?

Data = structured part + noise

We have noisy preference matrix $A \in \mathbb{R}^{m \times n}$

What about the missing data points?

Data = structured part + noise

We have noisy preference matrix $A \in \mathbb{R}^{m \times n}$
Structured part: low-rank

What about the missing data points?

Data = structured part + noise

We have noisy preference matrix $A \in \mathbb{R}^{m \times n}$
Structured part: low-rank
Noise: high-rank but spread out

What about the missing data points?

Data = structured part + noise

We have noisy preference matrix $A \in \mathbb{R}^{m \times n}$
Structured part: low-rank
Noise: high-rank but spread out
Idea: find best low-rank approximation (say rank 100)

Singular value decomposition

For every $A \in \mathbb{C}^{m \times n}$ its *singular value decomposition* is $A = U^{\dagger} \Sigma V$ where $U \in \mathbb{C}^{m \times m}$, $V \in \mathbb{C}^{n \times n}$ unitaries and $\Sigma \in \mathbb{R}^{m \times n}$ has non-zero elements only on the diagonal.

What about the missing data points?

Data = structured part + noise

We have noisy preference matrix $A \in \mathbb{R}^{m \times n}$
Structured part: low-rank
Noise: high-rank but spread out
Idea: find best low-rank approximation (say rank 100)

Singular value decomposition

For every $A \in \mathbb{C}^{m \times n}$ its *singular value decomposition* is $A = U^\dagger \Sigma V$ where $U \in \mathbb{C}^{m \times m}$, $V \in \mathbb{C}^{n \times n}$ unitaries and $\Sigma \in \mathbb{R}^{m \times n}$ has non-zero elements only on the diagonal.
We can also write $A = \sum_{i=1}^{m} \sigma_i |u_i\rangle\langle v_i|$, where u_i, v_i are the columns of U, V and $\sigma_1 \geq \sigma_2 \geq \ldots \geq \sigma_m \geq 0$ are the singular values of A.

What about the missing data points?

Data = structured part + noise

We have noisy preference matrix $A \in \mathbb{R}^{m \times n}$
Structured part: low-rank
Noise: high-rank but spread out
Idea: find best low-rank approximation (say rank 100)

Singular value decomposition

For every $A \in \mathbb{C}^{m \times n}$ its *singular value decomposition* is $A = U^\dagger \Sigma V$ where $U \in \mathbb{C}^{m \times m}$, $V \in \mathbb{C}^{n \times n}$ unitaries and $\Sigma \in \mathbb{R}^{m \times n}$ has non-zero elements only on the diagonal.
We can also write $A = \sum_{i=1}^{m} \sigma_i |u_i\rangle\langle v_i|$, where u_i, v_i are the columns of U, V and $\sigma_1 \geq \sigma_2 \geq \ldots \geq \sigma_m \geq 0$ are the singular values of A.

Fact: the best rank-k approximation of A is $\tilde{A} = \sum_{i=1}^{k} \sigma_i |u_i\rangle\langle v_i|$.
(Best in terms of the Frobenius norm: $\|M\|_F = \sqrt{\sum_{i,j} |M_{ij}|^2}$.)

We want to get a good recommendation

Given user i

Suppose we have a state proportional to $|A_{i.}\rangle$ (the i-th row of A).

We want to get a good recommendation

Given user i

Suppose we have a state proportional to $|A_{i.}\rangle$ (the i-th row of A).
We can write $|A_{i.}\rangle = \sum_{j=1}^{m} \alpha_j \left|v_j\right\rangle$, and we want $\left|\tilde{A}_{i.}\right\rangle = \sum_{j=1}^{k} \alpha_j \left|v_j\right\rangle$.

We want to get a good recommendation

Given user i

Suppose we have a state proportional to $|A_{i.}\rangle$ (the i-th row of A).
We can write $|A_{i.}\rangle = \sum_{j=1}^{m} \alpha_j \left|v_j\right\rangle$, and we want $\left|\tilde{A}_{i.}\right\rangle = \sum_{j=1}^{k} \alpha_j \left|v_j\right\rangle$.

Exercise 1: Prove that $A^\dagger A = \sum_{i=1}^{m} \sigma_i^2 |v_i\rangle\langle v_i|$

We want to get a good recommendation

Given user *i*

Suppose we have a state proportional to $|A_{i.}\rangle$ (the *i*-th row of *A*).
We can write $|A_{i.}\rangle = \sum_{j=1}^{m} \alpha_j \left|v_j\right\rangle$, and we want $\left|\tilde{A}_{i.}\right\rangle = \sum_{j=1}^{k} \alpha_j \left|v_j\right\rangle$.

Exercise 1: Prove that $A^\dagger A = \sum_{i=1}^{m} \sigma_i^2 |v_i\rangle\langle v_i|$

Quantum solution

Observe that $A^\dagger A = \sum_{i=1}^{m} \sigma_i^2 |v_i\rangle\langle v_i| \implies$ apply phase estimation with $A^\dagger A$ on $|A_{i.}\rangle$ to filter out small singular values. (For simplicity let's assume phase estimation works ideally.)

We want to get a good recommendation

Given user *i*

Suppose we have a state proportional to $|A_{i.}\rangle$ (the i-th row of A).
We can write $|A_{i.}\rangle = \sum_{j=1}^{m} \alpha_j \left|v_j\right\rangle$, and we want $\left|\tilde{A}_{i.}\right\rangle = \sum_{j=1}^{k} \alpha_j \left|v_j\right\rangle$.

Exercise 1: Prove that $A^\dagger A = \sum_{i=1}^{m} \sigma_i^2 |v_i\rangle\langle v_i|$

Quantum solution

Observe that $A^\dagger A = \sum_{i=1}^{m} \sigma_i^2 |v_i\rangle\langle v_i| \implies$ apply phase estimation with $A^\dagger A$ on $|A_{i.}\rangle$ to filter out small singular values. (For simplicity let's assume phase estimation works ideally.)

$\sum_{j=1}^{m} \alpha_j \left|v_j\right\rangle$

We want to get a good recommendation

Given user i

Suppose we have a state proportional to $|A_{i.}\rangle$ (the i-th row of A).
We can write $|A_{i.}\rangle = \sum_{j=1}^{m} \alpha_j \left|v_j\right\rangle$, and we want $\left|\tilde{A}_{i.}\right\rangle = \sum_{j=1}^{k} \alpha_j \left|v_j\right\rangle$.

Exercise 1: Prove that $A^\dagger A = \sum_{i=1}^{m} \sigma_i^2 |v_i\rangle\langle v_i|$

Quantum solution

Observe that $A^\dagger A = \sum_{i=1}^{m} \sigma_i^2 |v_i\rangle\langle v_i| \implies$ apply phase estimation with $A^\dagger A$ on $|A_{i.}\rangle$ to filter out small singular values. (For simplicity let's assume phase estimation works ideally.)

$\sum_{j=1}^{m} \alpha_j \left|v_j\right\rangle \mapsto \sum_{j=1}^{m} \alpha_j \left|v_j\right\rangle \left|\sigma_j^2\right\rangle$

We want to get a good recommendation

Given user i

Suppose we have a state proportional to $|A_{i.}\rangle$ (the i-th row of A).
We can write $|A_{i.}\rangle = \sum_{j=1}^{m} \alpha_j \left|v_j\right\rangle$, and we want $\left|\tilde{A}_{i.}\right\rangle = \sum_{j=1}^{k} \alpha_j \left|v_j\right\rangle$.

Exercise 1: Prove that $A^\dagger A = \sum_{i=1}^{m} \sigma_i^2 |v_i\rangle\langle v_i|$

Quantum solution

Observe that $A^\dagger A = \sum_{i=1}^{m} \sigma_i^2 |v_i\rangle\langle v_i| \implies$ apply phase estimation with $A^\dagger A$ on $|A_{i.}\rangle$ to filter out small singular values. (For simplicity let's assume phase estimation works ideally.)

$$\sum_{j=1}^{m} \alpha_j \left|v_j\right\rangle \mapsto \sum_{j=1}^{m} \alpha_j \left|v_j\right\rangle \left|\sigma_j^2\right\rangle \mapsto \sum_{j=1}^{m} \alpha_j \left|v_j\right\rangle \left|\sigma_j^2\right\rangle \left|\chi(\sigma_j^2 \geq \sigma^2)\right\rangle$$

We want to get a good recommendation

Given user i

Suppose we have a state proportional to $|A_{i.}\rangle$ (the i-th row of A).
We can write $|A_{i.}\rangle = \sum_{j=1}^{m} \alpha_j \left|v_j\right\rangle$, and we want $\left|\tilde{A}_{i.}\right\rangle = \sum_{j=1}^{k} \alpha_j \left|v_j\right\rangle$.

Exercise 1: Prove that $A^\dagger A = \sum_{i=1}^{m} \sigma_i^2 |v_i\rangle\langle v_i|$

Quantum solution

Observe that $A^\dagger A = \sum_{i=1}^{m} \sigma_i^2 |v_i\rangle\langle v_i| \implies$ apply phase estimation with $A^\dagger A$ on $|A_{i.}\rangle$ to filter out small singular values. (For simplicity let's assume phase estimation works ideally.)

$$\sum_{j=1}^{m} \alpha_j \left|v_j\right\rangle \mapsto \sum_{j=1}^{m} \alpha_j \left|v_j\right\rangle \left|\sigma_j^2\right\rangle \mapsto \sum_{j=1}^{m} \alpha_j \left|v_j\right\rangle \left|\sigma_j^2\right\rangle \left|\chi(\sigma_j^2 \geq \sigma^2)\right\rangle \mapsto \sum_{j=1}^{m} \alpha_j \left|v_j\right\rangle \underbrace{\left|\chi(\sigma_j^2 \geq \sigma^2)\right\rangle}_{\text{measure}}$$

We want to get a good recommendation

Given user *i*

Suppose we have a state proportional to $|A_{i.}\rangle$ (the i-th row of A).
We can write $|A_{i.}\rangle = \sum_{j=1}^{m} \alpha_j \left|v_j\right\rangle$, and we want $\left|\tilde{A}_{i.}\right\rangle = \sum_{j=1}^{k} \alpha_j \left|v_j\right\rangle$.

Exercise 1: Prove that $A^\dagger A = \sum_{i=1}^{m} \sigma_i^2 |v_i\rangle\langle v_i|$

Quantum solution

Observe that $A^\dagger A = \sum_{i=1}^{m} \sigma_i^2 |v_i\rangle\langle v_i| \Longrightarrow$ apply phase estimation with $A^\dagger A$ on $|A_{i.}\rangle$ to filter out small singular values. (For simplicity let's assume phase estimation works ideally.)

$$\sum_{j=1}^{m} \alpha_j \left|v_j\right\rangle \mapsto \sum_{j=1}^{m} \alpha_j \left|v_j\right\rangle \left|\sigma_j^2\right\rangle \mapsto \sum_{j=1}^{m} \alpha_j \left|v_j\right\rangle \left|\sigma_j^2\right\rangle \left|\chi(\sigma_j^2 \geq \sigma^2)\right\rangle \mapsto \sum_{j=1}^{m} \alpha_j \left|v_j\right\rangle \underbrace{\left|\chi(\sigma_j^2 \geq \sigma^2)\right\rangle}_{\textit{measure}}$$

If we get outcome 1 the state is proportional to $\left|\tilde{A}_{i.}\right\rangle$.

We want to get a good recommendation

Given user i

Suppose we have a state proportional to $|A_{i.}\rangle$ (the i-th row of A).
We can write $|A_{i.}\rangle = \sum_{j=1}^{m} \alpha_j \left|v_j\right\rangle$, and we want $\left|\tilde{A}_{i.}\right\rangle = \sum_{j=1}^{k} \alpha_j \left|v_j\right\rangle$.

Exercise 1: Prove that $A^\dagger A = \sum_{i=1}^{m} \sigma_i^2 |v_i\rangle\langle v_i|$

Quantum solution

Observe that $A^\dagger A = \sum_{i=1}^{m} \sigma_i^2 |v_i\rangle\langle v_i| \implies$ apply phase estimation with $A^\dagger A$ on $|A_{i.}\rangle$ to filter out small singular values. (For simplicity let's assume phase estimation works ideally.)

$$\sum_{j=1}^{m} \alpha_j \left|v_j\right\rangle \mapsto \sum_{j=1}^{m} \alpha_j \left|v_j\right\rangle \left|\sigma_j^2\right\rangle \mapsto \sum_{j=1}^{m} \alpha_j \left|v_j\right\rangle \left|\sigma_j^2\right\rangle \left|\chi(\sigma_j^2 \geq \sigma^2)\right\rangle \mapsto \sum_{j=1}^{m} \alpha_j \left|v_j\right\rangle \underbrace{\left|\chi(\sigma_j^2 \geq \sigma^2)\right\rangle}_{\textit{measure}}$$

If we get outcome 1 the state is proportional to $\left|\tilde{A}_{i.}\right\rangle$.
Measuring the state then gives recommendation j with probability $\propto |\tilde{A}_{ij}|^2$.

Major difficulty: how to input the data?

Data conversion: classical to quantum

- Given $b \in \mathbb{R}^m$ prepare

$$|b\rangle = \sum_{i=1}^{m} \frac{b_i}{||b||} |i\rangle$$

Major difficulty: how to input the data?

Data conversion: classical to quantum

- Given $b \in \mathbb{R}^m$ prepare

$$|b\rangle = \sum_{i=1}^{m} \frac{b_i}{||b||} |i\rangle$$

- Given $A \in \mathbb{R}^{m \times n}$ construct quantum circuit (block-encoding)

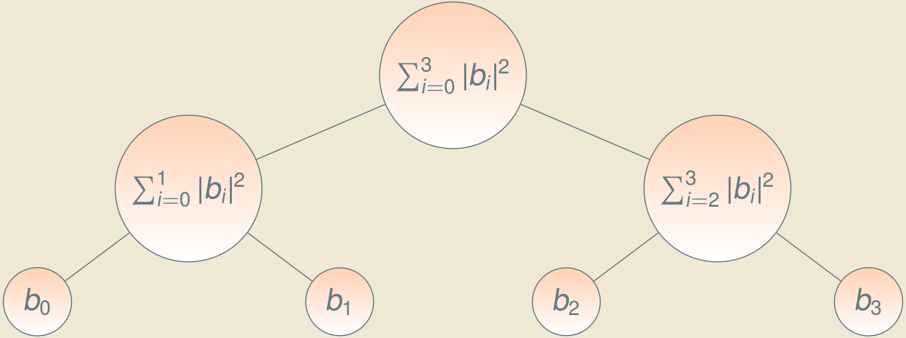
$$U = \begin{pmatrix} A / \, ||A||_F & \cdot \\ \cdot & \cdot \end{pmatrix}.$$

How to preserve the exponential advantage?

Solution: assume QRAM (readable in superposition)

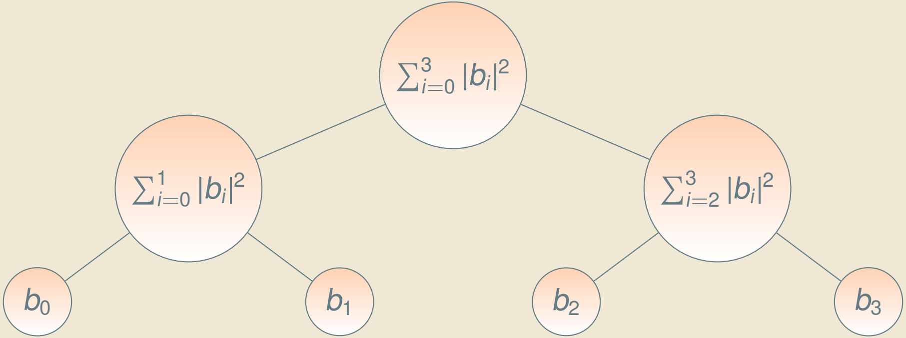
Solution: assume QRAM (readable in superposition)

Data structure (for simplicity let us assume $\|b\| = 1$)



Solution: assume QRAM (readable in superposition)

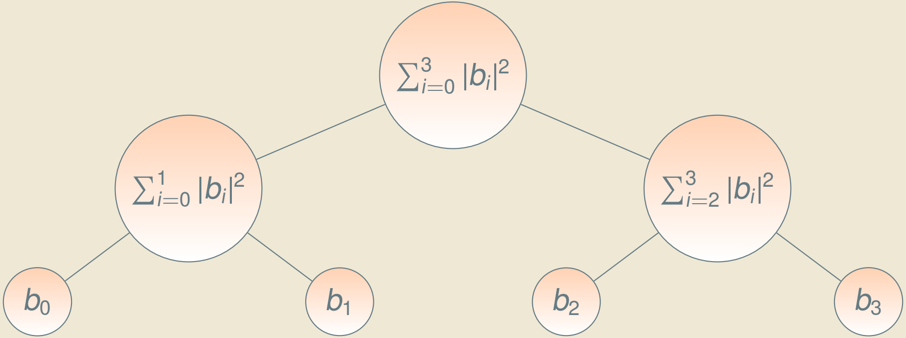
Data structure (for simplicity let us assume $\|b\| = 1$)



First prepare: $\sqrt{\sum_{i=0}^{1} |b_i|^2}|0\rangle + \sqrt{\sum_{i=2}^{3} |b_i|^2}|1\rangle$

Solution: assume QRAM (readable in superposition)

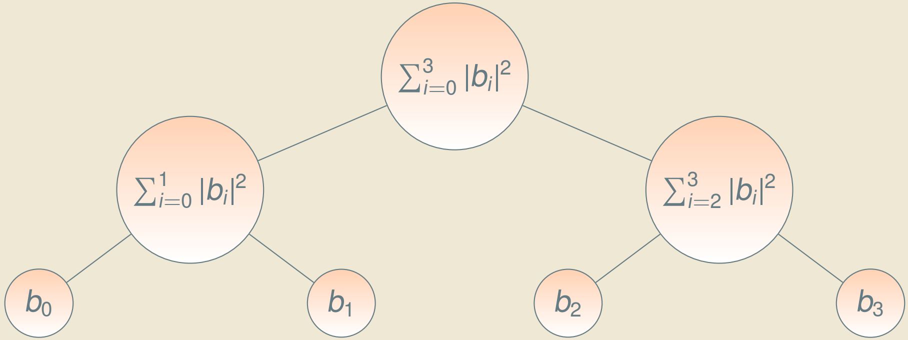
Data structure (for simplicity let us assume $\|b\| = 1$)



First prepare: $\sqrt{\sum_{i=0}^{1} |b_i|^2}|0\rangle + \sqrt{\sum_{i=2}^{3} |b_i|^2}|1\rangle$ – use rotation gate $\begin{pmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{pmatrix}$

Solution: assume QRAM (readable in superposition)

Data structure (for simplicity let us assume $\|b\| = 1$)

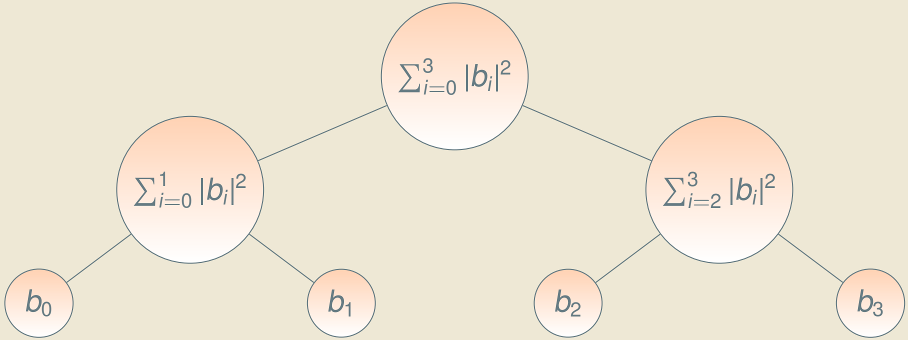


First prepare: $\sqrt{\sum_{i=0}^{1} |b_i|^2}|0\rangle + \sqrt{\sum_{i=2}^{3} |b_i|^2}|1\rangle$ – use rotation gate $\begin{pmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{pmatrix}$

Map $\sqrt{\sum_{i=0}^{1} |b_i|^2}|00\rangle \mapsto |b_0||00\rangle + |b_1||01\rangle$ and

Solution: assume QRAM (readable in superposition)

Data structure (for simplicity let us assume $\|b\| = 1$)

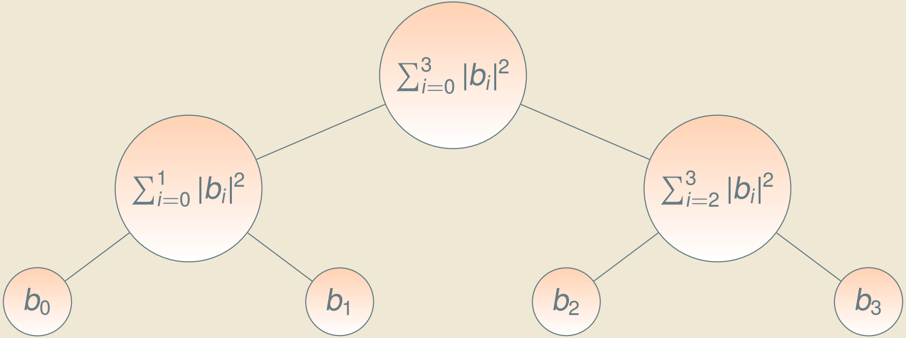


First prepare: $\sqrt{\sum_{i=0}^{1} |b_i|^2}|0\rangle + \sqrt{\sum_{i=2}^{3} |b_i|^2}|1\rangle$ – use rotation gate $\begin{pmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{pmatrix}$

Map $\sqrt{\sum_{i=0}^{1} |b_i|^2}|00\rangle \mapsto |b_0||00\rangle + |b_1||01\rangle$ and $\sqrt{\sum_{i=2}^{3} |b_i|^2}|10\rangle \mapsto |b_2||10\rangle + |b_3||11\rangle$

Solution: assume QRAM (readable in superposition)

Data structure (for simplicity let us assume $\|b\| = 1$)



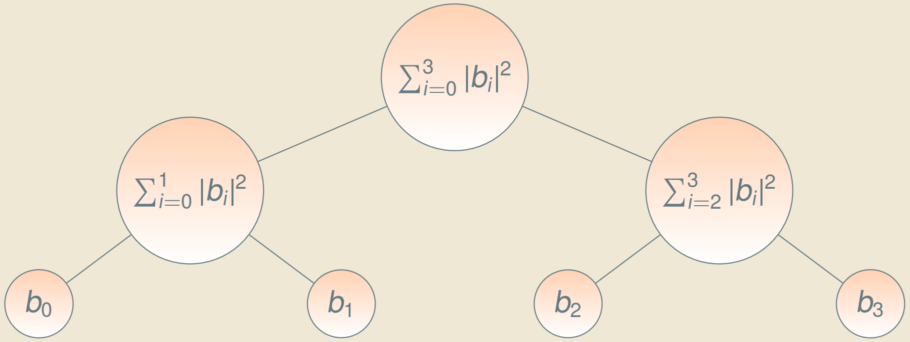
First prepare: $\sqrt{\sum_{i=0}^{1} |b_i|^2}|0\rangle + \sqrt{\sum_{i=2}^{3} |b_i|^2}|1\rangle$ – use rotation gate $\begin{pmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{pmatrix}$

Map $\sqrt{\sum_{i=0}^{1} |b_i|^2}|00\rangle \mapsto |b_0||00\rangle + |b_1||01\rangle$ and $\sqrt{\sum_{i=2}^{3} |b_i|^2}|10\rangle \mapsto |b_2||10\rangle + |b_3||11\rangle$

Add phases to get $b_0|00\rangle + b_1|01\rangle + b_2|10\rangle + b_3|11\rangle$

On-line updates to the data structure

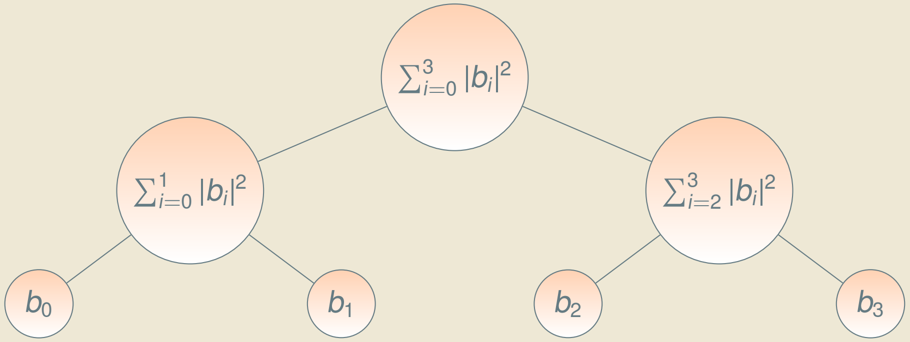
Data structure



–Update an entry and then its parent, grand parent, etc.

On-line updates to the data structure

Data structure



–Update an entry and then its parent, grand parent, etc.

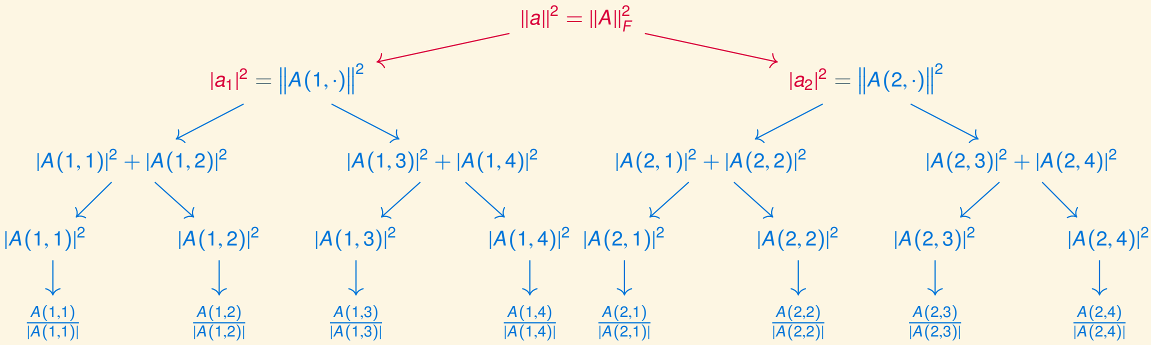
Cost is about the depth: $\log(\text{dimension})$

Data structure for the matrix A

Let a be the vector of row norms such that $a_i = \|A_{i.}\|$.

Data structure for the matrix A

Let a be the vector of row norms such that $a_i = \|A_{i.}\|$.



Dynamic data structure for a matrix $A \in \mathbb{C}^{2\times 4}$. We compose the data structure for a with the data structure for A's rows.

Quantum algorithms

Exercise 2: Let $R : |0\rangle|i\rangle \mapsto \frac{|A_{i.}\rangle|i\rangle}{\|A_{i.}\|}$ and $C : |0\rangle|j\rangle \mapsto \frac{|j\rangle|a\rangle}{\|a\|}$.
Show that $U = R^\dagger C$ is a block-encoding of $A / \|A\|_F$.

Quantum algorithms

Exercise 2: Let $R: |0\rangle|i\rangle \mapsto \frac{|A_{i.}\rangle|i\rangle}{\|A_{i.}\|}$ and $C: |0\rangle|j\rangle \mapsto \frac{|j\rangle|a\rangle}{\|a\|}$.
Show that $U = R^\dagger C$ is a block-encoding of $A/\|A\|_F$.
Exercise 3: Show that $U^\dagger(2|0\rangle\langle 0| - I)U$ is a block-encoding of $2\frac{A^\dagger A}{\|A\|_F^2} - I$.

Quantum algorithms

Exercise 2: Let $R: |0\rangle|i\rangle \mapsto \frac{|A_{i.}\rangle|i\rangle}{\|A_{i.}\|}$ and $C: |0\rangle|j\rangle \mapsto \frac{|j\rangle|a\rangle}{\|a\|}$.
Show that $U = R^\dagger C$ is a block-encoding of $A / \|A\|_F$.
Exercise 3: Show that $U^\dagger(2|0\rangle\langle 0| - I)U$ is a block-encoding of $2\frac{A^\dagger A}{\|A\|_F^2} - I$.

Recommendation systems

Given i prepare quantum state $|A_{i.}\rangle / \|A_{i.}\|$ ($\log(m+n)$ QRAM calls). Then prepare $\left|\tilde{A}_{i.}\right\rangle$ by phase estimation to precision $\frac{\sigma^2}{\|A\|_F^2}$ and then a measurement, the cost is

$$\widetilde{O}\left(\frac{\|A\|_F^2}{\sigma^2}\right)$$

Quantum algorithms

Exercise 2: Let $R: |0\rangle|i\rangle \mapsto \frac{|A_{i.}\rangle|i\rangle}{\|A_{i.}\|}$ and $C: |0\rangle|j\rangle \mapsto \frac{|j\rangle|a\rangle}{\|a\|}$.
Show that $U = R^\dagger C$ is a block-encoding of $A / \|A\|_F$.
Exercise 3: Show that $U^\dagger(2|0\rangle\langle 0| - I)U$ is a block-encoding of $2\frac{A^\dagger A}{\|A\|_F^2} - I$.

Recommendation systems

Given i prepare quantum state $|A_{i.}\rangle / \|A_{i.}\|$ ($\log(m+n)$ QRAM calls). Then prepare $\left|\tilde{A}_{i.}\right\rangle$ by phase estimation to precision $\frac{\sigma^2}{\|A\|_F^2}$ and then a measurement, the cost is

$$\widetilde{O}\left(\frac{\|A\|_F^2}{\sigma^2}\right) \quad \text{times post-selection cost factor:} \quad \frac{\|A_{i.}\|^2}{\left\|\tilde{A}_{i.}\right\|^2}$$

Quantum algorithms

Exercise 2: Let $R: |0\rangle|i\rangle \mapsto \frac{|A_{i.}\rangle|i\rangle}{\|A_{i.}\|}$ and $C: |0\rangle|j\rangle \mapsto \frac{|j\rangle|a\rangle}{\|a\|}$.
Show that $U = R^\dagger C$ is a block-encoding of $A/\|A\|_F$.
Exercise 3: Show that $U^\dagger(2|0\rangle\langle 0| - I)U$ is a block-encoding of $2\frac{A^\dagger A}{\|A\|_F^2} - I$.

Recommendation systems

Given i prepare quantum state $|A_{i.}\rangle / \|A_{i.}\|$ ($\log(m+n)$ QRAM calls). Then prepare $|\tilde{A}_{i.}\rangle$ by phase estimation to precision $\frac{\sigma^2}{\|A\|_F^2}$ and then a measurement, the cost is

$$\widetilde{O}\left(\frac{\|A\|_F^2}{\sigma^2}\right) \quad \text{times post-selection cost factor:} \quad \frac{\|A_{i.}\|^2}{\|\tilde{A}_{i.}\|^2}$$

Tomorrow we will see

This can be improved quadratically!

Surely exponential speed-up compared to classical, right?

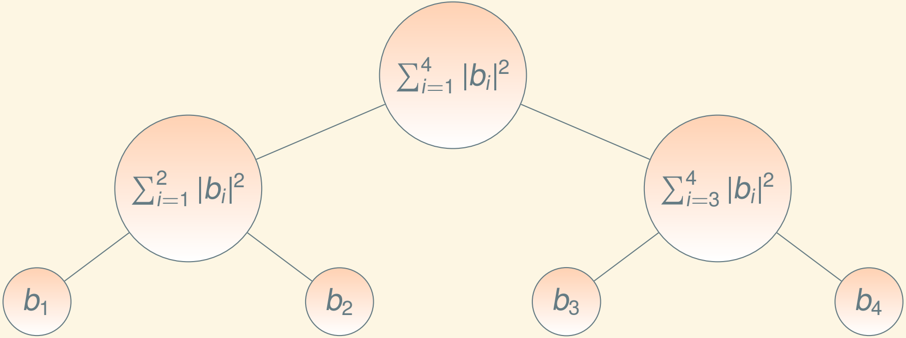
2018:

Ewin Tang
Quantum Recommendation Systems
arXiv:1807.04271
Ewin Tang
arXiv:1811.00414
Quantum PCA
Ewin Tang
arXiv:1811.04909
Low rank HHL

2018:

Image source: Quantum Computing Memes for QMA-Complete Teens

Sampling form the input vectors?

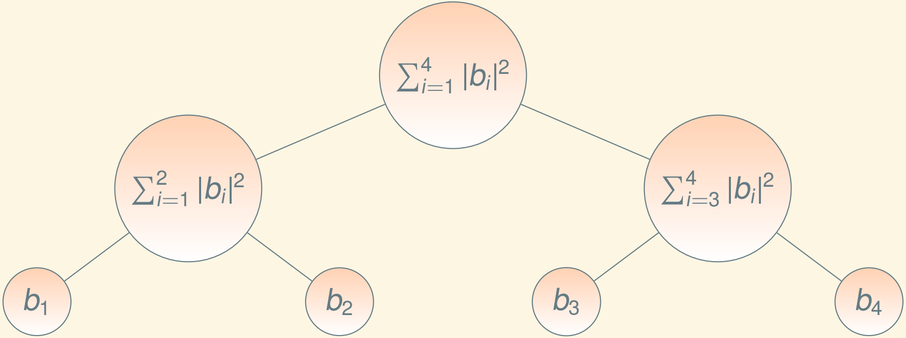


Data structure for storing $b \in \mathbb{R}^m$

If stored in (classical) RAM, in time $\mathcal{O}(\log(\text{dimension}))$ we can

- query b_i, and

Sampling form the input vectors?



Data structure for storing $b \in \mathbb{R}^m$

If stored in (classical) RAM, in time $\mathcal{O}(\log(\text{dimension}))$ we can

- query b_i, and
- sample i distributed $\propto |b_i|^2$, and

Computing inner products

Computing $\langle x, y \rangle$ for normalized vectors x, y

If we have sample and query access to x and query access to y

- sample i distributed $\propto |x_i|^2$, and output $\frac{y_i}{x_i}$

Computing inner products

Computing $\langle x, y\rangle$ for normalized vectors x, y

If we have sample and query access to x and query access to y

- sample i distributed $\propto |x_i|^2$, and output $\frac{y_i}{x_i}$
- $\mathbb{E} = \sum_{i=1}^{n} \left|x_i\right|^2 \frac{y_i}{x_i} = \langle x, y\rangle$;

Computing inner products

Computing $\langle x, y \rangle$ for normalized vectors x, y

If we have sample and query access to x and query access to y

- sample i distributed $\propto |x_i|^2$, and output $\frac{y_i}{x_i}$
- $\mathbb{E} = \sum_{i=1}^n \left|x_i\right|^2 \frac{y_i}{x_i} = \langle x, y \rangle$; $\qquad \mathbb{E}|\cdot|^2 = \sum_{i=1}^n \left|x_i\right|^2 \left|\frac{y_i}{x_i}\right|^2 = \|y\|^2 = 1$

Computing inner products

Computing $\langle x, y \rangle$ for normalized vectors x, y

If we have sample and query access to x and query access to y

- sample i distributed $\propto |x_i|^2$, and output $\frac{y_i}{x_i}$
- $\mathbb{E} = \sum_{i=1}^n \left|x_i\right|^2 \frac{y_i}{x_i} = \langle x, y \rangle$; $\qquad \mathbb{E}|\cdot|^2 = \sum_{i=1}^n \left|x_i\right|^2 \left|\frac{y_i}{x_i}\right|^2 = \|y\|^2 = 1$

Computing matrix elements

If we have sample and query access to A and query access to x, y

- We want to compute $x^T A y$

Computing inner products

Computing $\langle x, y\rangle$ for normalized vectors x, y

If we have sample and query access to x and query access to y

- sample i distributed $\propto |x_i|^2$, and output $\frac{y_i}{x_i}$
- $\mathbb{E} = \sum_{i=1}^n \left|x_i\right|^2 \frac{y_i}{x_i} = \langle x, y\rangle;$ $\qquad \mathbb{E}|\cdot|^2 = \sum_{i=1}^n \left|x_i\right|^2 \left|\frac{y_i}{x_i}\right|^2 = \|y\|^2 = 1$

Computing matrix elements

If we have sample and query access to A and query access to x, y

- We want to compute $x^T A y = \mathrm{Tr}(x^T A y) = \mathrm{Tr}(A y x^T)$

Computing inner products

Computing $\langle x, y\rangle$ for normalized vectors x, y

If we have sample and query access to x and query access to y

- sample i distributed $\propto |x_i|^2$, and output $\frac{y_i}{x_i}$
- $\mathbb{E} = \sum_{i=1}^{n} \left|x_i\right|^2 \frac{y_i}{x_i} = \langle x, y\rangle$; $\qquad \mathbb{E}|\cdot|^2 = \sum_{i=1}^{n} \left|x_i\right|^2 \left|\frac{y_i}{x_i}\right|^2 = \|y\|^2 = 1$

Computing matrix elements

If we have sample and query access to A and query access to x, y

- We want to compute $x^T A y = \mathrm{Tr}(x^T A y) = \mathrm{Tr}(A y x^T) = \langle A, y x^T\rangle_{HS}$

Low rank approximation of $A^\dagger A$

$A^\dagger A = \sum_{i=1}^{m} |A_{i.}\rangle\langle A_{i.}|$

With probability $\frac{\|A_{i.}\|^2}{\|A\|_F^2} = \frac{|a_i|^2}{\|a\|^2}$ sample i and output the rank-1 matrix $\|A\|_F^2 \cdot \frac{|A_{i.}\rangle\langle A_{i.}|}{\|\|A_{i.}\|^2\|}$.

Low rank approximation of $A^\dagger A$

$A^\dagger A = \sum_{i=1}^m |A_{i.}\rangle\langle A_{i.}|$

With probability $\frac{\|A_{i.}\|^2}{\|A\|_F^2} = \frac{|a_i|^2}{\|a\|^2}$ sample i and output the rank-1 matrix $\|A\|_F^2 \cdot \frac{|A_{i.}\rangle\langle A_{i.}|}{\left\|\|A_{i.}\|^2\right\|}$.
The expectation value is

$$\sum_i p_i \|A\|_F^2 \cdot \frac{|A_{i.}\rangle\langle A_{i.}|}{\left\|\|A_{i.}\|^2\right\|} = \sum_i \frac{\|A_{i.}\|^2}{\|A\|_F^2} \|A\|_F^2 \cdot \frac{|A_{i.}\rangle\langle A_{i.}|}{\left\|\|A_{i.}\|^2\right\|} = \sum_{i=1}^m |A_{i.}\rangle\langle A_{i.}| = A^\dagger A$$

Low rank approximation of $A^\dagger A$

$A^\dagger A = \sum_{i=1}^m |A_{i.}\rangle\langle A_{i.}|$

With probability $\frac{\|A_{i.}\|^2}{\|A\|_F^2} = \frac{|a_i|^2}{\|a\|^2}$ sample i and output the rank-1 matrix $\|A\|_F^2 \cdot \frac{|A_{i.}\rangle\langle A_{i.}|}{\left\|\|A_{i.}\|^2\right\|}$.
The expectation value is

$$\sum_i p_i \|A\|_F^2 \cdot \frac{|A_{i.}\rangle\langle A_{i.}|}{\left\|\|A_{i.}\|^2\right\|} = \sum_i \frac{\|A_{i.}\|^2}{\|A\|_F^2} \|A\|_F^2 \cdot \frac{|A_{i.}\rangle\langle A_{i.}|}{\left\|\|A_{i.}\|^2\right\|} = \sum_{i=1}^m |A_{i.}\rangle\langle A_{i.}| = A^\dagger A$$

Each random matrix has norm $\|A\|_F^2$.

Low rank approximation of $A^\dagger A$

$A^\dagger A = \sum_{i=1}^m |A_{i.}\rangle\langle A_{i.}|$

With probability $\frac{\|A_{i.}\|^2}{\|A\|_F^2} = \frac{|a_i|^2}{\|a\|^2}$ sample i and output the rank-1 matrix $\|A\|_F^2 \cdot \frac{|A_{i.}\rangle\langle A_{i.}|}{\|\|A_{i.}\|^2\|}$.
The expectation value is

$$\sum_i p_i \|A\|_F^2 \cdot \frac{|A_{i.}\rangle\langle A_{i.}|}{\|\|A_{i.}\|^2\|} = \sum_i \frac{\|A_{i.}\|^2}{\|A\|_F^2} \|A\|_F^2 \cdot \frac{|A_{i.}\rangle\langle A_{i.}|}{\|\|A_{i.}\|^2\|} = \sum_{i=1}^m |A_{i.}\rangle\langle A_{i.}| = A^\dagger A$$

Each random matrix has norm $\|A\|_F^2$.

Matrix Chernoff bound – Ahlswede & Winter (2000), Tropp (2010)

Let $B \in R^{n\times n}$ and suppose that $\mathbb{E}[X] = B$, and $\|X - B\| \le \gamma$.
If $X_1, X_2, \ldots$ are iid copies of X, then

$$\mathbb{P}\left(\left\|B - \frac{1}{t}\sum_{i=1}^t X_i\right\| > \varepsilon\right) \le 2n \exp\left(-\frac{\varepsilon^2 t}{3\gamma^2}\right).$$

Working with small linear combinations

$$y = x^{(1)} + x^{(2)}.$$

(Rejection) sample from the linear combination $x^{(1)} + x^{(1)}$

If we have sample and query access to $x^{(1)}, x^{(2)}$

- sample ℓ distributed as $\left|x^{(\ell)}\right|^2 / \left(\left\|x^{(1)}\right\|^2 + \left\|x^{(2)}\right\|^2\right)$, then

Working with small linear combinations

$$y = x^{(1)} + x^{(2)}.$$

(Rejection) sample from the linear combination $x^{(1)} + x^{(1)}$

If we have sample and query access to $x^{(1)}, x^{(2)}$

- sample ℓ distributed as $\left|x^{(\ell)}\right|^2 / \left(\left\|x^{(1)}\right\|^2 + \left\|x^{(2)}\right\|^2\right)$, then
- sample i distributed as $\left|x_i^{(\ell)}\right|^2$, and accept with probability

$$\left|x_i^{(1)} + x_i^{(2)}\right|^2 \Big/ \left(\left|x_i^{(1)}\right|^2 + \left|x_i^{(2)}\right|^2\right)$$

Working with small linear combinations

$$y = x^{(1)} + x^{(2)}.$$

(Rejection) sample from the linear combination $x^{(1)} + x^{(1)}$

If we have sample and query access to $x^{(1)}, x^{(2)}$

- sample ℓ distributed as $\left|x^{(\ell)}\right|^2 / \left(\left\|x^{(1)}\right\|^2 + \left\|x^{(2)}\right\|^2\right)$, then
- sample i distributed as $\left|x_i^{(\ell)}\right|^2$, and accept with probability

$$\left|x_i^{(1)} + x_i^{(2)}\right|^2 \Big/ \left(\left|x_i^{(1)}\right|^2 + \left|x_i^{(2)}\right|^2\right)$$

We see i with probability $\left|x_i^{(1)} + x_i^{(2)}\right|^2 \Big/ \left(\left\|x^{(1)}\right\|^2 + \left\|x^{(2)}\right\|^2\right)$.

Working with small linear combinations

$$y = x^{(1)} + x^{(2)}.$$

(Rejection) sample from the linear combination $x^{(1)} + x^{(1)}$

If we have sample and query access to $x^{(1)}, x^{(2)}$

- sample ℓ distributed as $\left|x^{(\ell)}\right|^2 / \left(\left\|x^{(1)}\right\|^2 + \left\|x^{(2)}\right\|^2\right)$, then
- sample i distributed as $\left|x_i^{(\ell)}\right|^2$, and accept with probability

$$\left|x_i^{(1)} + x_i^{(2)}\right|^2 \Big/ \left(\left|x_i^{(1)}\right|^2 + \left|x_i^{(2)}\right|^2\right)$$

We see i with probability $\left|x_i^{(1)} + x_i^{(2)}\right|^2 \Big/ \left(\left\|x^{(1)}\right\|^2 + \left\|x^{(2)}\right\|^2\right)$. Total acceptance prob.:

$$\sum_{i=1}^{n} \mathbb{P}(\text{Output } i)$$

Working with small linear combinations

$$y = x^{(1)} + x^{(2)}.$$

(Rejection) sample from the linear combination $x^{(1)} + x^{(1)}$

If we have sample and query access to $x^{(1)}, x^{(2)}$

- sample ℓ distributed as $\left|x^{(\ell)}\right|^2 / \left(\left\|x^{(1)}\right\|^2 + \left\|x^{(2)}\right\|^2\right)$, then
- sample i distributed as $\left|x_i^{(\ell)}\right|^2$, and accept with probability

$$\left|x_i^{(1)} + x_i^{(2)}\right|^2 \Big/ \left(\left|x_i^{(1)}\right|^2 + \left|x_i^{(2)}\right|^2\right)$$

We see i with probability $\left|x_i^{(1)} + x_i^{(2)}\right|^2 \Big/ \left(\left\|x^{(1)}\right\|^2 + \left\|x^{(2)}\right\|^2\right)$. Total acceptance prob.:

$$\sum_{i=1}^{n} \mathbb{P}(\text{Output } i) = \left|x_i^{(1)} + x_i^{(2)}\right|^2 \Big/ \left(\left\|x^{(1)}\right\|^2 + \left\|x^{(2)}\right\|^2\right)$$

Working with small linear combinations

$$y = x^{(1)} + x^{(2)}.$$

(Rejection) sample from the linear combination $x^{(1)} + x^{(1)}$

If we have sample and query access to $x^{(1)}, x^{(2)}$

- sample ℓ distributed as $\left|x^{(\ell)}\right|^2 / \left(\left\|x^{(1)}\right\|^2 + \left\|x^{(2)}\right\|^2\right)$, then
- sample i distributed as $\left|x_i^{(\ell)}\right|^2$, and accept with probability

$$\left|x_i^{(1)} + x_i^{(2)}\right|^2 \Big/ \left(\left|x_i^{(1)}\right|^2 + \left|x_i^{(2)}\right|^2\right)$$

We see i with probability $\left|x_i^{(1)} + x_i^{(2)}\right|^2 \Big/ \left(\left\|x^{(1)}\right\|^2 + \left\|x^{(2)}\right\|^2\right)$. Total acceptance prob.:

$$\sum_{i=1}^{n} \mathbb{P}(\text{Output } i) = \left|x_i^{(1)} + x_i^{(2)}\right|^2 \Big/ \left(\left\|x^{(1)}\right\|^2 + \left\|x^{(2)}\right\|^2\right) = \frac{\|y\|^2}{\left\|x^{(1)}\right\|^2 + \left\|x^{(2)}\right\|^2}$$

Overall complexity

Punchline

No exponential speed-ups!

Overall complexity

Punchline

No exponential speed-ups!
Widely applicable, e.g., recommendation systems, low-rank matrix inversion, etc.

Overall complexity

Punchline

No exponential speed-ups!
Widely applicable, e.g., recommendation systems, low-rank matrix inversion, etc.

Complexity comparison

$\widetilde{\mathcal{O}}\left(\frac{\|A\|_F}{\sigma}\right)$ quantum vs. $\widetilde{\mathcal{O}}\left(\frac{\|A\|_F^6\|A\|^{10}}{\sigma^{16}\varepsilon^6}\right)$ classical
Are quantum Big Data algorithms doomed now???

Overall complexity

Punchline

No exponential speed-ups!
Widely applicable, e.g., recommendation systems, low-rank matrix inversion, etc.

Complexity comparison

$\widetilde{O}\left(\frac{\|A\|_F}{\sigma}\right)$ quantum vs. $\widetilde{O}\left(\frac{\|A\|_F^6\|A\|^{10}}{\sigma^{16}\varepsilon^6}\right)$ classical
Are quantum Big Data algorithms doomed now???

Open questions

Better classical algorithms? Better quantum algorithms?

Is there hope for a genuine quantum speedup?

Topological data analysis: Lloyd, Garnerone, and Zanardi (2016),

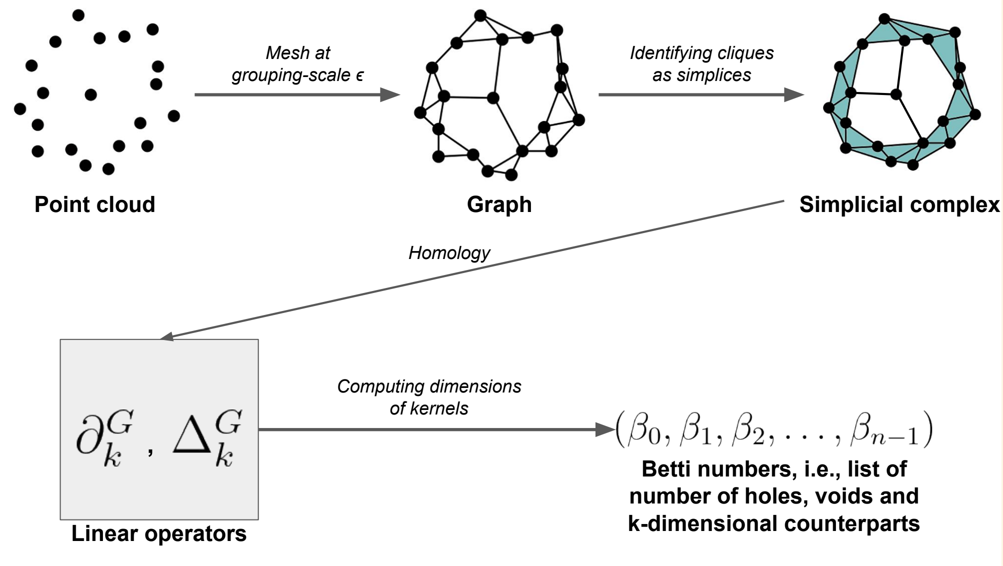


Image from Gyurik, Cade, Dunjko arXiv:2005.02607 (2020)

Zero-sum games (van Apeldoorn, G – arXiv: 1904.03180)

Pay-off matrix of Alice is $A \in \mathbb{R}^{m \times n}$. Expected pay-off for strategies x, y: $x^T A y$

Zero-sum games (van Apeldoorn, G – arXiv: 1904.03180)

Pay-off matrix of Alice is $A \in \mathbb{R}^{m \times n}$. Expected pay-off for strategies x, y: $x^T A y$

Fictitious play for approximate Nash-equilibrium (Grigoriadis & Khachiyan 1995)

Start with $x^{(0)} \leftarrow 0 \in \mathbb{R}^m$ and $y^{(0)} \leftarrow 0 \in \mathbb{R}^n$

Zero-sum games (van Apeldoorn, G – arXiv: 1904.03180)

Pay-off matrix of Alice is $A \in \mathbb{R}^{m\times n}$. Expected pay-off for strategies x, y: $x^T A y$

Fictitious play for approximate Nash-equilibrium (Grigoriadis & Khachiyan 1995)

Start with $x^{(0)} \leftarrow 0 \in \mathbb{R}^m$ and $y^{(0)} \leftarrow 0 \in \mathbb{R}^n$
for $t = 1, 2, \ldots, \widetilde{\mathcal{O}}\left(\frac{1}{\varepsilon^2}\right)$ **do**

- $P^{(t)} \leftarrow e^{-A^T x^{(t)}}$ and $Q^{(t)} \leftarrow e^{A y^{(t)}}$
- $p^{(t)} \leftarrow P^{(t)} / \left\|P^{(t)}\right\|_1$ and $q^{(t)} \leftarrow Q^{(t)} / \left\|Q^{(t)}\right\|_1$
- Sample $a \sim p^{(t)}$ and $b \sim q^{(t)}$
- $y^{(t+1)} = y^{(t)} + \frac{\varepsilon}{4} e_a$ and $x^{(t+1)} = x^{(t)} + \frac{\varepsilon}{4} e_b$

Zero-sum games (van Apeldoorn, G – arXiv: 1904.03180)

Pay-off matrix of Alice is $A \in \mathbb{R}^{m \times n}$. Expected pay-off for strategies x, y: $x^T A y$

Fictitious play for approximate Nash-equilibrium (Grigoriadis & Khachiyan 1995)

Start with $x^{(0)} \leftarrow 0 \in \mathbb{R}^m$ and $y^{(0)} \leftarrow 0 \in \mathbb{R}^n$
for $t = 1, 2, \ldots, \widetilde{O}\left(\frac{1}{\varepsilon^2}\right)$ **do**

- $P^{(t)} \leftarrow e^{-A^T x^{(t)}}$ and $Q^{(t)} \leftarrow e^{A y^{(t)}}$
- $p^{(t)} \leftarrow P^{(t)} / \left\|P^{(t)}\right\|_1$ and $q^{(t)} \leftarrow Q^{(t)} / \left\|Q^{(t)}\right\|_1$
- Sample $a \sim p^{(t)}$ and $b \sim q^{(t)}$
- $y^{(t+1)} = y^{(t)} + \frac{\varepsilon}{4} e_a$ and $x^{(t+1)} = x^{(t)} + \frac{\varepsilon}{4} e_b$

The main task is Gibbs sampling from a linear-combination of vectors.

Quantum rejection sampling: Ozols, Rötteler, Roland '11

The main task is Gibbs sampling ($\propto e^{Ay^{(t)}}$) from a linear-combination of vectors.

Quantum rejection sampling: Ozols, Rötteler, Roland ’11

The main task is Gibbs sampling ($\propto e^{Ay^{(t)}}$) from a linear-combination of vectors.

Idea: quantum rejection sampling.

Quantum rejection sampling: Ozols, Rötteler, Roland '11

The main task is Gibbs sampling ($\propto e^{Ay^{(t)}}$) from a linear-combination of vectors.
Idea: quantum rejection sampling.

- Compute largest entry c of $Ay^{(t)}$ in time: $\mathcal{O}\left(\sqrt{m}/\varepsilon^2\right)$

Quantum rejection sampling: Ozols, Rötteler, Roland ’11

The main task is Gibbs sampling ($\propto e^{Ay^{(t)}}$) from a linear-combination of vectors.
Idea: quantum rejection sampling.

- Compute largest entry c of $Ay^{(t)}$ in time: $\mathcal{O}\left(\sqrt{m}/\varepsilon^2\right)$
- Sample $i \in [m]$ with probability $\frac{1}{m}$, accept with probability $e^{A_{i.}y^{(t)}-c}$.

Quantum rejection sampling: Ozols, Rötteler, Roland ’11

The main task is Gibbs sampling ($\propto e^{Ay^{(t)}}$) from a linear-combination of vectors.
Idea: quantum rejection sampling.

- Compute largest entry c of $Ay^{(t)}$ in time: $\mathcal{O}\left(\sqrt{m}/\varepsilon^2\right)$
- Sample $i \in [m]$ with probability $\frac{1}{m}$, accept with probability $e^{A_{i.}y^{(t)}-c}$.
- Repeat $\mathcal{O}\left(\sqrt{m}\right)$-times $\Rightarrow$ complexity $\widetilde{\mathcal{O}}\left((\sqrt{n}+\sqrt{m})/\varepsilon^4\right)$

Quantum rejection sampling: Ozols, Rötteler, Roland ’11

The main task is Gibbs sampling ($\propto e^{Ay^{(t)}}$) from a linear-combination of vectors.
Idea: quantum rejection sampling.

- Compute largest entry c of $Ay^{(t)}$ in time: $\mathcal{O}\left(\sqrt{m}/\varepsilon^2\right)$
- Sample $i \in [m]$ with probability $\frac{1}{m}$, accept with probability $e^{A_{i.}y^{(t)}-c}$.
- Repeat $\mathcal{O}\left(\sqrt{m}\right)$-times $\Rightarrow$ complexity $\widetilde{\mathcal{O}}\left((\sqrt{n}+\sqrt{m})/\varepsilon^4\right)$

Exercise 4: work out the details of the above algorithm

Quantum rejection sampling: Ozols, Rötteler, Roland ’11

The main task is Gibbs sampling ($\propto e^{Ay^{(t)}}$) from a linear-combination of vectors.
Idea: quantum rejection sampling.

- Compute largest entry c of $Ay^{(t)}$ in time: $\mathcal{O}\left(\sqrt{m}/\varepsilon^2\right)$
- Sample $i \in [m]$ with probability $\frac{1}{m}$, accept with probability $e^{A_{i.}y^{(t)}-c}$.
- Repeat $\mathcal{O}\left(\sqrt{m}\right)$-times $\Rightarrow$ complexity $\widetilde{\mathcal{O}}\left((\sqrt{n}+\sqrt{m})/\varepsilon^4\right)$

Exercise 4: work out the details of the above algorithm
(Note: Can be improved to $\widetilde{\mathcal{O}}\left((\sqrt{n}+\sqrt{m})/\varepsilon^3\right)$ by using approximate counting.)

LPs ($\sim$ zero-sum games) and SDPs

A generalization of *Linear programs (LPs)*.

LPs ($\sim$ zero-sum games) and SDPs

A generalization of *Linear programs (LPs)*. Let $x \in \mathbb{R}^n$

$$\mathrm{OPT} = \min \quad \langle c, x \rangle$$

LPs ($\sim$ zero-sum games) and SDPs

A generalization of *Linear programs (LPs)*. Let $x \in \mathbb{R}^n$

$$\begin{aligned} \mathrm{OPT} = \min \quad & \langle c, x \rangle \\ \text{s.t.} \quad & \langle a_j, x \rangle \leq b_j \quad \text{for all } j \in [m], \\ & x \geq 0 \end{aligned}$$

LPs ($\sim$ zero-sum games) and SDPs

A generalization of *Linear programs (LPs)*. Let $X \in \mathbb{R}^{n \times n}$

$$\begin{aligned} \mathrm{OPT} = \min \quad & \langle C, X \rangle \\ \text{s.t.} \quad & \langle A_j, X \rangle \leq b_j \quad \text{for all } j \in [m], \\ & X \geq 0 \end{aligned}$$

LPs ($\sim$ zero-sum games) and SDPs

A generalization of *Linear programs (LPs)*. Let $X \in \mathbb{R}^{n \times n}$

$$\begin{aligned} \mathrm{OPT} = \min \quad & \mathrm{Tr}(CX) \\ \text{s.t.} \quad & \mathrm{Tr}(A_j X) \leq b_j \quad \text{for all } j \in [m], \\ & X \geq 0 \end{aligned}$$

Assumptions and formalization

LPs ($\sim$ zero-sum games) and SDPs

A generalization of *Linear programs (LPs)*. Let $X \in \mathbb{R}^{n \times n}$

$$\begin{aligned} \mathrm{OPT} = \min \quad & \mathrm{Tr}(CX) \\ \text{s.t.} \quad & \mathrm{Tr}(A_j X) \le b_j \quad \text{for all } j \in [m], \\ & X \ge 0 \end{aligned}$$

Assumptions and formalization

- $n \times n$ variable matrix X, with m constraints.

LPs ($\sim$ zero-sum games) and SDPs

A generalization of *Linear programs (LPs)*. Let $X \in \mathbb{R}^{n \times n}$

$$\begin{aligned} \mathrm{OPT} = \min \quad & \mathrm{Tr}(CX) \\ \text{s.t.} \quad & \mathrm{Tr}(A_j X) \leq b_j \quad \text{for all } j \in [m], \\ & X \succeq 0 \end{aligned}$$

Assumptions and formalization

- $n \times n$ variable matrix X, with m constraints.
- Assume $\|C\|, \|A_j\| \leq 1$ and s-sparse.

LPs ($\sim$ zero-sum games) and SDPs

A generalization of *Linear programs (LPs)*. Let $X \in \mathbb{R}^{n \times n}$

$$
\begin{aligned}
\mathrm{OPT} = \min \quad & \mathrm{Tr}(CX) \\
\text{s.t.} \quad & \mathrm{Tr}(A_j X) \leq b_j \quad \text{for all } j \in [m], \\
& X \succeq 0
\end{aligned}
$$

Assumptions and formalization

- $n \times n$ variable matrix X, with m constraints.
- Assume $\|C\|, \|A_j\| \leq 1$ and s-sparse.
- A priori known bounds $\mathrm{Tr}[X] \leq R$ and $\sum_{j=0}^{m} y_j \leq r$.

LPs ($\sim$ zero-sum games) and SDPs

A generalization of *Linear programs (LPs)*. Let $X \in \mathbb{R}^{n \times n}$

$$\begin{aligned} \mathrm{OPT} = \min \quad & \mathrm{Tr}(CX) \\ \text{s.t.} \quad & \mathrm{Tr}(A_j X) \leq b_j \quad \text{for all } j \in [m], \\ & X \succeq 0 \end{aligned}$$

Assumptions and formalization

- $n \times n$ variable matrix X, with m constraints.
- Assume $\|C\|, \|A_j\| \leq 1$ and s-sparse.
- A priori known bounds $\mathrm{Tr}[X] \leq R$ and $\sum_{j=0}^{m} y_j \leq r$.
- Goal: additive ε-approximation of the optimum.

LPs ($\sim$ zero-sum games) and SDPs

A generalization of *Linear programs (LPs)*. Let $X \in \mathbb{R}^{n \times n}$

$$
\begin{aligned}
\mathrm{OPT} = \min \quad & \mathrm{Tr}(CX) \\
\text{s.t.} \quad & \mathrm{Tr}(A_j X) \leq b_j \quad \text{for all } j \in [m], \\
& X \succeq 0
\end{aligned}
$$

Assumptions and formalization

- $n \times n$ variable matrix X, with m constraints.
- Assume $\|C\|, \|A_j\| \leq 1$ and s-sparse.
- A priori known bounds $\mathrm{Tr}[X] \leq R$ and $\sum_{j=0}^{m} y_j \leq r$.
- Goal: additive ε-approximation of the optimum.

Examples: MAXCUT, Lovász theta number,
Sum-of-Squares, General Adversary bound, ...

LPs ($\sim$ zero-sum games) and SDPs

A generalization of *Linear programs (LPs)*. Let $X \in \mathbb{R}^{n\times n}$

$$\begin{aligned} \mathrm{OPT} = \min \quad & \mathrm{Tr}(CX) \\ \text{s.t.} \quad & \mathrm{Tr}(A_j X) \leq b_j \quad \text{for all } j \in [m], \\ & X \succeq 0 \end{aligned}$$

Assumptions and formalization

- $n \times n$ variable matrix X, with m constraints.
- Assume $\|C\|, \|A_j\| \leq 1$ and s-sparse.
- A priori known bounds $\mathrm{Tr}[X] \leq R$ and $\sum_{j=0}^{m} y_j \leq r$.
- Goal: additive ε-approximation of the optimum.

Examples: MAXCUT, Lovász theta number,
Sum-of-Squares, General Adversary bound, ...
Brandão et al., van Apeldoorn et al. 2016-18 quantum solver $\widetilde{O}\Big((\sqrt{n} + \sqrt{m})(Rr/\varepsilon)^5\Big)$

Learning from quantum data

Quantum principal component analysis (PCA)

Suppose as input we get a copy of a quantum state $|\psi_i\rangle$ with probability p_i.

- The mixed input quantum state is $\rho = \sum_i p_i |\psi_i\rangle\langle\psi_i|$
- (For simplicity let us assume $\langle\psi_i, \psi_j\rangle = \delta_{ij}$)
- $\mathcal{O}\left(t^2/\varepsilon\right)$ copies enable implementing ε-approximately $e^{it\rho}$ see "Quantum principal component analysis" by Lloyd, Mohseni, Rebentrost (2013) [Exercise 5: 18.7]
- Using phase estimation we can mark the input states $|\psi_i\rangle|0\rangle \mapsto |\psi_i\rangle|p_i\rangle$

Advantage with quantum memory

Without quantum memory at least $\sim 2^{n/2}$ experiments are needed to learn a fixed property of the principal component of an unknown n-qubit quantum state, while a constant number of experiments suffice when two copies can be jointly processed.

Quantum advantage in learning from experiments: **Huang**, Broughton, Cotler, Chen, Li, Mohseni, Neven, Babbush, Kueng, Preskill, McClean (2021)